CLAIMS

What is claimed is:

1. A novel architecture for set associative cache, comprising:

a set associative cache having a plurality of ways wherein the ways are segmented into a plurality of banks and wherein a first way has a fast access time;

access control logic which manages access to the cache and is coupled to said plurality of ways;

a plurality of muxes coupled to said first way in each of said banks and coupled to said access control logic; and

wherein the access control logic controls the mux in a bank to remap any defective way in a bank to the first way in that same bank.

2. The architecture of claim 1 wherein said first way has a faster access time because it has a physically shorter path to said access control logic.

3. The architecture of claim 1 further comprising self test logic coupled to said access control logic to test the cache for defects.

4. The architecture of claim 3 wherein said self test logic tests the cache for defects on power up.

5. The architecture of claim 3 wherein said self test logic stores the location of defects in a status register.

6. The architecture of claim 5 wherein said access control logic reads the location of defects in the cache from the status register to determine proper control of said muxes.

7. The architecture of claim 1 wherein said set associative cache has a data array having a plurality of ways wherein the ways are segmented into a plurality of banks and wherein a first way has a faster access time.

8. The architecture of claim 1 comprising a plurality of ways having a fast access time and a plurality of muxes coupled to said plurality of ways in each of said banks and coupled to said access control logic.

9. The architecture of claim 8 wherein the access control logic controls the plurality of muxes in a bank to remap any defective way in a bank to a different way in that same bank.

10. The architecture of claim 1 wherein the access time of said first way (t_1) is sufficiently fast such that the added time of the mux (t_{mux}) will not add any latency.

11. The architecture of claim 10 wherein the access time of said first way (t_1) added to the time of the mux (t_{mux}) is less than or equal to the access time of the slowest way (t_n).

12. The architecture of claim 10 wherein the access time of said first way (t_1) added to the time of the mux (t_{mux}) is less than or equal to a system clock cycle (t_{clk}).

13. A microprocessor die, comprising:

self test logic which tests the die for defects;

a set associative cache having a plurality of ways wherein the ways are segmented into a plurality of banks;

access control logic which manages access to the cache coupled to said self test logic and coupled to said plurality of ways in said cache;

a first way in said cache which has a physically shorter path to said access control logic;

a plurality of muxes coupled to said first way in each of said plurality of banks and coupled to said access control logic; and

wherein the access control logic controls the mux in a bank to remap any defective way in a bank to the first way in that same bank.

14. The microprocessor die of claim 13 comprising a plurality of ways having a physically shorter path to said access control logic and a plurality of muxes coupled to said plurality of ways in each of said banks and coupled to said access control logic.

15. The microprocessor die of claim 14 wherein the access control logic controls the plurality of muxes in a bank to remap any defective way in a bank to a different way in that same bank.

16. The microprocessor die of claim 13 wherein the access time of said first way (t_1) is sufficiently fast such that the added time of the mux (t_{mux})will not add any latency to the microprocessor.

17. The microprocessor die of claim 13 wherein the access time of said first way (t_1) added to the time of the mux (t_{mux}) is less than or equal to the access time of the slowest way (t_n).

18. The microprocessor die of claim 13 wherein the access time of said first way (t_1) added to the time of the mux (t_{mux}) is less than or equal to a system clock cycle (t_{clk}).

19. A method of absorbing defects in a set associative cache, comprising:

providing a set associative cache with a plurality of ways wherein the ways are segmented into a plurality of banks and wherein a first way has a fast access time;

providing a plurality of muxes coupled to said first way in each of said banks; and

using the mux in a bank to remap any defective way in a bank to the first way in that same bank.

20. The method of claim 19 further comprising the step of testing for errors in the cache.

21. The method of claim 19 further comprising the step of disabling a way in a bank when that way is defective.

22. The method of claim 19 comprising a plurality of ways having a fast access time and a plurality of muxes coupled to said plurality of ways in each of said banks.

23. The method of claim 22 wherein the plurality of muxes in a bank are used to remap any defective way in a bank to a different way in that same bank.

24. A computer system, comprising:

a power supply;

a microprocessor comprising:

a set associative cache having a plurality of ways wherein the ways are segmented into a plurality of banks;

access control logic which manages access to the cache coupled to said plurality of ways in said cache;

a first way in said cache which has a physically shorter path to said access control logic;

a plurality of muxes coupled to said first way in each of said plurality of banks and coupled to said access control logic; and

wherein the access control logic can control the mux in a bank to remap any defective way in a bank to the first way in that same bank.